Problema 3 - Sistema antidoping

Glauber da Silva Santana

Engenharia de Computação – Universidade Estadual de Feira de Santana (UEFS)
CEP 44036-900 – Feira de Santana – BA – Brasil

Departamento de Ciências Exatas e Tecnologia

glauber.silva14@gmail.com

1. Introdução

É a administração ilícita de uma droga estimulante com intuito de suprimir temporariamente a fadiga, aumentar ou diminuir a velocidade, melhorar ou piorar a atuação de um animal ou esportista. A comissão médica do comitê olímpico internacional instituiu durante os jogos olímpicos do México (1968) a aplicação de testes anti-dopping sistemáticos, decidindo que seriam excluídos dos jogos os atletas comprovadamente dopados.

Por isto foi solicitado aos alunos de Engenharia de Computação o desenvolvimento de um programa que fosse capaz de cadastrar essas jogadores para o exame antidoping da Copa do Mundo da Rússia, que acontece agora, no ano de 2018. O sistema deve permitir o inclusão e remoção de jogadores, alteração das informações cadastradas, sorteio para o exame, e filtragem dos mesmos.

No desenvolvimento desta solução que será apresentada foram utilizados a Linguagem de Programação C, como, funções de entrada e saída, estruturas de dados simples, métodos condicionais, laços de repetição e ferramentas para manipulação de memória. Esta solução está completa, permitindo que todas as ações sejam feitas, no entanto, pela forma implementa ao se utilizar da entrada de texto o programa entra em loop e a saída de texto não gera informações, sendo assim, todos os dados são impressos na tela do terminal (prompt). Neste relatório serão abordados os seguintes tópicos: metodologia, resultados e discussões e conclusão.

2. Metodologia

Durante as sessões tutoriais não houveram discussões. Com base no conhecimento prévio adquirido durante os outros tutoriais não houveram grandes dificuldades para implementação desta solução, a novidade aqui foi a utilização da função para randomizar os dados.

- Funções.
    1. Funcionamento da função srand(time()), esta é a função que nos permite a randomização para a rand(), ela “implanta” uma semente tendo como base o horário do sistema operacional tornando assim possível a randomização de diferentes valores.
    2. Funcionamento da função rand(), sem a utilização da função “auxiliar” srand(time()), rand() irá nos entregar um número pseudo-randômico, pois, assim que há a primeira chamada para esta função, ela irá obter um

valor, o qual toda chamada consecutiva a esta função retornará este mesmo valor.

Para cumprir com as exigências do problema, foram utilizadas a biblioteca padrão da linguagem C determinada com três arquivos de cabeçalho: *stdio.h*, *stdlib.h*, *string.h, stdbool.h.* Os quais possibilitaram a utilização de funções para apresentar e ler o conteúdo, controlar processos e realizar conversões, manipular matrizes e registros de dados, e pré-definição de *MACROS* para utilização no programa, assim como a implementação de sub-rotinas para modularização.

O algoritmo foi elaborado no software Atom para o sistema operacional Windows, onde também foi testado.

2.1 Sub-rotinas ou Funções

Partindo do estudo da linguagem, funções foram utilizadas, o algoritmos está bem modularizado, partindo do princípio de que uma sub-rotina como esta deve ter um objetivo e realizar apenas uma tarefa, foram criadas então métodos para menu, cadastro, coleta de dados, verificação de informação, tratamento de erros e para os cálculos do programa.

Cada função tem sua respectiva tarefa sendo executada separadamente conforme seja solicitado, podendo assim, uma determinada função chamar uma outra para "solucionar um problema" naquela situação.

2.2 Registros (*structs)*

Os registros de dados ou (*structs*), são estruturas que podem armazenar variáveis de tipos diferentes em um só lugar, armazenando espaço na memória para que todos estes valores possam ser guardados. Sendo assim, foi criada uma struct para representar os jogadores *(representadas abaixo),* e seus atributos como: nome, ano de nascimento, país em que joga, posição em que joga e número da sua camisa. Equipe, com nome, participantes e variáveis de identificação.

```
typedef struct Player {
  bool  is_sorted;
  int id;
  char name[MAX];
  int  birth_year;
  char nation[20];
  int  shirt_number;
  char position[20];
}player_t;
```

Imagem 1-Demonstração dos registros de dados (structs)
Fonte: Próprio autor

Como pode-se observar na imagem acima, foi utilizada a função "typedef struct" para redefinir o tipo básico struct, permitindo chamá-la pelo apelido "player_t*"*.

2.3 Lista encadeada

Visto que devemos cadastrar “n” jogadores esta solução se utiliza de listas encadeadas, o que nos permite a criação teoricamente infinita de posições para armazenar quantos jogadores nos for necessário.

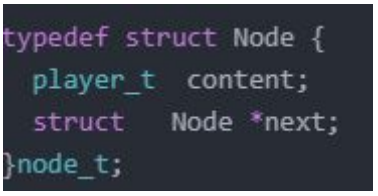

Imagem 2-Demonstração da estrutura (struct) da Lista Encadeada
Fonte: Próprio autor

2.4 Ponteiros

Bem, ponteiro ou apontador, é um tipo de dado que guarda um endereço de memória de uma outra variável e não um valor em si. Com a utilização de funções, há a necessidade de se acessar os valores de variáveis em diferentes partes de um programa, onde estes poderiam, ou não serem alterados, sendo assim, devido a esta necessidade utiliza-se ponteiros para que valores de uma variável possam ser modificados dentro de uma função na qual esta não foi declarada.

3. Resultados e Discussões

Decidi fazer o programa em inglês, pois isto me ajuda no aprendizado e proporciona uma melhor visualização das informações no terminal sem a presença dos caracteres especiais.

Ao iniciar o programa o usuário encontrará algumas opções:

Selecionando a (OPÇÃO 1) o usuário poderá registrar um jogador.

Selecionando a (OPÇÃO 2) o usuário poderá modificar a camisa de um jogador.

Selecionando a (OPÇÃO 3) o usuário poderá remover um jogador do sistema.

Selecionando a (OPÇÃO 4) o usuário poderá visualizar todos os jogadores de determinada posição

Selecionando a (OPÇÃO 5)

Selecionando a (OPÇÃO 6)

Selecionando a (OPÇÃO 7)

Selecionando a (OPÇÃO 8)

3.1 Considerações

Os objetivos propostos pelo problema foram alcançados.

Bem, neste problema para mim, particularmente não há muito o que comentar, eu já havia trabalhado com as listas antes e todo conhecimento utilizado aqui eu vim trabalhando durante os outros dois problemas que nos foram apresentados.

Eu realmente esperava que as sessões tutoriais fossem render muito nesta fase, visto que este é um dos assuntos que mais geram dúvida e que, dessa forma as sessões pudessem ser divertidas, cheias de perguntas e falhas, erros, que nos trariam até aqui com êxito, com todos tendo aprendido alguma coisa nova durante as discussões nas sessões, ou ainda apresentando muitas soluções diferentes. Por exemplo, em uma outra sessão tutorial vi uma solução que consistia em utilizar uma segunda lista encadeada para armazenar os jogadores sorteados.

No entanto, isso não se tornou verdade em nossos encontros e logo após a segunda para terceira sessão houve um desânimo bem maior que o esperado, minha solução já estava implementada, mas eu desejava que alguém tivesse dúvidas, como tinha sido na primeira sessão, o que também acabou não ocorrendo, então, apenas decidi por esperar a data de entrega do problema.

4. Conclusões

O programa tinha como objetivo principal a criação de um sistema que pudesse cadastrar jogadores para um sistema de antidoping.
Este foi um problema interessante, graças ao exercício das outras duas soluções anteriores pude encontrar um caminho mais rápido para esta solução, creio que poderia ser melhor explorado, talvez com um sistema de jogos, onde poderíamos ter times, jogos entre estes, substituições que pudessem ser feitas, um sistema diferente com opções diferentes.